e-ISSN 2444-8729

https://doi.org/10.14201/eks20171843551

How to Improve Computational Thinking: a Case Study

Cómo mejorar el pensamiento computacional: un estudio de caso

José Alberto Quitério Figueiredo

Research Unit for Inland Development, Polytechnic of Guarda 6300-559 Guarda, Portugal jfig@ipg.pt
https://orcid.org/0000-0002-8501-1686

Abstract

One of the best skills for everyone, for now, and for the future, is problem-solving. Computational thinking is the way to help us to develop that skill. Computational Thinking can be defined as a set of skills for problem-solving based on computer techniques. Computational thinking is needed everywhere and is going to be a key to success in almost all careers, not only for a scientist but for many professionals, like doctors, lawyers, teachers or farmers. For many problems it is a good idea to make a plan for its resolution using some of the techniques of computer science, such as: breaking down a complex problem into smaller parts that are more manageable and easier to understand, or solve—decomposition; looking for similarities among and within problems and others experiences—pattern recognition; focusing on the important information only, and pulling out specific differences to make one solution work for multiple problems: abstraction; developing a step-by-step solution to the problem: algorithms. This plan can be used by everyone, regardless of their area of knowledge, task or age. It is essential that these techniques are practiced and developed very early. In recent years we have to see the proliferation of numerous projects with the specific objective of encouraging the study of Computational thinking. The projects of massification of computational thinking and coding are now starting to be implemented in our education system in Portugal. Most students of the first year of the Computer Engineering course, from the IPG, mostly did not have the opportunity to develop computational thinking throughout their student life. In this paper, we present the results of a case study using follow and give instructions to improve their capacities in Computational Thinking..

Keywords

Computational Thinking; Programming; CS0; CS1; Learning Programming; Teaching Programming

Resumen

Una de las mejores habilidades para todos, por ahora y para el futuro, es la resolución de problemas. El pensamiento computacional es la manera de ayudarnos a desarrollar esa habilidad. El pensamiento computacional se puede definir como un conjunto de habilidades para la resolución de problemas basadas en técnicas informáticas. El pensamiento computacional es necesario en todas partes y será la clave del éxito en casi todas las carreras, no solo para un científico, sino también para muchos profesionales, como médicos, abogados, docentes o agricultores. Para muchos problemas, es una buena idea hacer un plan para su resolución utilizando algunas de las técnicas de la informática, tales como: descomponer un problema complejo en partes más pequeñas que sean más manejables y fáciles de entender, o resolver—descomposición; buscando similitudes entre y dentro de problemas y otras experiencias—reconocimiento de patrones; centrándose solamente en la información importante y eliminando las diferencias específicas para que una solución funcione para múltiples problemas: abstracción; desarrollando una solución paso a paso al problema: algoritmos. Este plan puede ser utilizado por todos, independientemente de su área de conocimiento, tarea o edad. Es esencial que estas técnicas se practiquen y desarrollen muy temprano. Los proyectos de masificación del pensamiento computacional y la codificación están comenzando a implementarse en nuestro sistema educativo en Portugal. La mayoría de los estudiantes del primer año del curso de Ingeniería Informática, del IPG, en su mayoría no tuvieron la oportunidad de desarrollar el pensamiento computacional a lo largo de su vida estudiantil. En este artículo, presentamos los resultados de un estudio de caso usando instrucciones "da y sigue" para mejorar sus capacidades en el pensamiento computacional.

Palabras clave

Pensamiento computacional; Programación; CS0; CS1; Aprendiendo programación; Enseñando programación

Recepción: 14-11-2017 Revisión: 24-11-2017 Aceptación: 29-11-2017 Publicación: 31-12-2017

1. Computational Thinking an Overview

Computational thinking is a fundamental skill for everyone. The term computational thinking was made popular by Jeannette M. Wing (2006). The author defends the mass diffusion of computational thinking, just like reading, writing, and arithmetic. In recent years, we have witnessed the proliferation of projects sponsored by both governmental and non-governmental organizations, with the specific objective of encouraging computational thinking and the study of programming, especially in pre-university years.

Computational Thinking is an aptitude that allows us to create solutions to problems by making use of computer techniques (García-Peñalvo & Cruz-Benito, 2016; García-Peñalvo, 2016c). Computers can be used to solve problems. However, before a problem can be tackled, the problem itself and the ways in which it could be solved need to be understood. Computational thinking allows us to take a complex problem, understand what it is and develop possible solutions. We can then present these solutions in a way that a computer, a human being, or both, can understand. For many of the daily life tasks, from the simplest to the most complex, it is a good idea to make a plan for its resolution using some of the techniques of computer science, such as: breaking down a complex problem into smaller parts that are more manageable and easier to understand, or solve—decomposition; looking for similarities among and within problems and others experiences—pattern recognition; focusing on the important information only, and pulling out specific differences to make one solution work for multiple problems—abstraction; developing a step-by-step solution to the problem—algorithms. This plan can be used by everyone, regardless of their area of knowledge, task or age.

Computational Thinking is essential to the development of computer applications, but it can also be used to support problem solving across all disciplines, including math, science, and the humanities. Students who learn Computational Thinking across the syllabus can begin to see a relationship between subjects as well as between school and life outside the classroom (Pinto-Llorente, Casillas-Martín, Cabezas-González & García-Peñalvo, 2017).

Computational thinking teaches us to think, to find ways to solve a problem, to organize and plan the resolution of a task, and teaches us and gives us the courage, the methods and techniques to solve complex problems.

In this paper, we highlight the importance of computational thinking throughout the training of all students, especially in their pre-university education. In fact, most students who come to university have never had the opportunity to develop these skills of computational thinking. Programming is one of the best ways to develop those skills. In the first part of our work we present some of the tools

that help to motivate and encourage the taste for programming, especially those with resources to help teachers. In the second part of this paper, we present the results of a study involving a group of students of Computer Engineering from the Polytechnic of Guarda, Portugal. In this study, we explore the concepts of following and giving instruction and map design, in an attempt to improve students' computational thinking skills

2. Programming the Way to Computational Thinking-Tools

Science, Technology, Engineering and Mathematics (STEM) permeate nearly every facet of modern life and hold the key to solving many of humanity's most pressing current and future challenges. The United States' position in the global economy is declining, in part because U.S. workers lack fundamental knowledge in these fields. To address the critical issues of U.S. competitiveness and to better prepare the workforce, a Framework for K-12 Science Education proposes a new approach to K-12 science education that will capture students' interest and provide them with the necessary foundational knowledge in the field (National Research Council, 2012). This is just an example, but many other countries feel the need to promote the work to achieve this, by making it easier for teachers and others involved in STEM education, like the United Kingdom (STEM, n.d.).

It is known that there will be 1.4 million job openings for computing-related jobs by 2020, but at the current rate of people being prepared for those positions, only approximately 30% of them will be filled (Israel, Wherfel, Pearson, Shehab & Tapia, 2015). Employment of computer and information technology is projected to grow 12 percent from 2014 to 2024, faster than the average for all jobs (by U.S. Bureau of Labor Statistics, Office of Occupational Statistics and Employment Projections). In addition, there are several instructional benefits for students that result from the inclusion of computing within K-12 programs (Israel, et al., 2015). These include, among others, building higher-order thinking skills and increasing positive attitudes about computer science and computer science skills.

Despite the general attention on computer science, many teachers don't have professional skills in computer science or computational thinking. This is probably the reason why in recent years we have witnessed the proliferation of numerous projects with the specific objective of encouraging the study of programming, especially in pre-university years. Many organizations are working hard to set young people up for success in a digital world. In the work presented in (García-Peñalvo, Reimann, et al., 2016), an exhaustive research and respective evaluation of the existing tools for the teaching and learning of the programming for pre-university studies was made. Now, we present some of these tools and add others, especially those that allow teachers and parents to show students the best way:

- TACCLE 3. TACCLE 3 - Coding is a European Project (Taccle 3 – Supporting primary teachers to teach coding, n.d.) that supports primary school and other teachers who want to teach Computing to 4-14

year olds. TACCLE 3 is a project that provides practical ideas and the knowledge that teachers can use and the materials they need for introducing computing or coding in their classrooms (García-Peñalvo, 2016a, 2016b; García-Peñalvo, Hughes, et al., 2016).

- Scratch. Scratch is a programming language and an online community where everyone can program and share interactive media such as stories, games, and animation with people from all over the world. As children create with Scratch, they learn to think creatively, work collaboratively, and reason systematically. These are essential skills for life in the 21st century. Scratch is designed and maintained by the Lifelong Kindergarten group at the MIT Media Lab. (Scratch - Imagine, Program, Share, n.d.).

- Alice. Alice is an innovative block-based programming environment that makes it easy to create animations, build interactive narratives, or program simple games in 3D. Alice is designed to teach logical and computational thinking skills, fundamental principles of programming and to be a first exposure to object-oriented programming. The Alice Project provides supplemental tools and materials for teaching, which when used in diverse and demotivated groups has already given proven benefits in captivating and retaining those groups (Alice – Tell Stories. Build Games. Learn to Program., n.d.).

- Code.org. Launched in 2013, Code.org is a non-profit dedicated to expanding participation in computer science by making it available in more schools, and increasing participation by women and underrepresented students of colour. Their vision is that every student in every school should have the opportunity to learn computer science. They believe computer science and computer programming should be part of the core curriculum in education, alongside other science, technology, engineering, and mathematics (STEM) courses, such as biology, physics, chemistry and algebra.

The 'Hour of Code' is a global initiative by Computer Science Education Week (Computer Science Education Week, n.d.) and Code.org to introduce millions of students to one hour of computer science and computer programming.

- Khan Academy. Khan Academy offers practice exercises, instructional videos, and a personalized learning dashboard that empower learners to study at their own pace in and outside of the classroom, in diverse subjects: maths, science and engineering, computing, arts and humanities, economics and finance. Their mission is to provide a free world-class education for anyone, anywhere (Khan Academy | Free Online Courses, Lessons & Practice, n.d.).

- CS Unplugged. Their principles are: No Computers Required. Real Computer Science. Learning by doing. Fun. No specialized equipment. Variations encouraged. For everyone. Co-operative. Stand-alone Activities. Resilient (Computer Science Unplugged, n.d.).

- Tynker. Tynker is a revolutionary way to learn coding. Kids learn fundamental programming concepts with visual blocks, then progress to JavaScript and Python as they develop skills and gain confidence in their new abilities (Coding for Kids | Tynker, n.d.).

- Lightbot. In Lightbot, students must guide a robot to light up all the blue tiles in each level or puzzles. To do so, you must 'program' the robot using a set of instructions. Students may play the game in the Browser, or on Android or iOS devices. The main goal is to understand how to create and give a computer a set of instructions to follow (Lightbot, n.d.).

- Barefoot. Barefoot supports primary educators with the confidence, knowledge, skills and resources to teach computer science. Resources aligned to the curricular for all UK nations. This includes lesson plans and workshops, all designed to help teachers gain confidence in bringing computer science to life in the classroom ("Home - Barefoot Computing Barefoot Computing", n.d.). Barefoot Computing is supported by British telecommunications (BT) ("techliteracy", n.d.) commitment to help build a culture of tech literacy and Computing At School (CAS) (Computing At School, n.d.).

- CodeCombat. CodeCombat is a coding game that uses real typed code and personalized learning to teach computer science. CodeCombat uses typed code, not draganddrop blocks. They believe that getting students to real typed code as quickly as possible is critical to learning essential computer science concepts. It also allows far more creativity and flexibility students are free to solve problems however they see fit. CodeCombat is for teachers too, empowering them to use it in their classrooms, being this one of the goals of project (CodeCombat - Learn how to code by playing a game, n.d.).

- Kodable. Written by elementary school teachers, the Kodable curriculum makes coding for kids in their class possible. It focuses on excellent instruction with group and independent practice activities that build creativity, communication, and collaboration. Their goal is to reach all students and see computer science become part of a complete elementary education (Programming for Kids | Kodable, n.d.).

- MIT App Inventor. MIT App Inventor is an intuitive, visual programming environment that allows everyone – even schoolchildren – to build fully functional apps for smartphones and tablets. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to transition from being consumers of technology to becoming creators of it (MIT App Inventor, n.d.). App Inventor for Educators is an educational community. It is intended as a common online area to share ideas, resources, and find answers to questions (App Inventor for Educators – MIT App Inventor Educators Community, n.d.).

- LiveCode. LiveCode has a vision that everyone can code. They made the open source platform for building native mobile, desktop and server applications. The visual workflow allows the user to develop

apps with a simple and powerful programming language that empowers students to develop their computer science skills (LiveCode Ltd, n.d.).

- Touch Develop. Touch Develop is created in a friendly environment. With Touch Develop we can create apps on our mobile phone, tablets, and PCs, and share the apps you create in the cloud. Creative Coding Through Games And Apps is an introduction to computer science course built with Touch Develop. Includes day-by-day plans for implementing the curriculum as a 6, 9, 12, or an 18-week instructor-led course(Microsoft Touch Develop - create apps everywhere, on all your devices!, n.d.).

- Blockly. Blockly is library that adds a visual code editor to web and Android apps. The Blockly editor uses interlocking, graphical blocks to represent code concepts like variables, logical expressions, loops, and more. It allows users to apply programming principles without having to worry about syntax or the intimidation of a blinking cursor on the command line. They have developed a range of resources, programs, scholarships, and grant opportunities to engage students and educators around the world interested in computer science (Blockly | Google Developers, n.d.).

- Snap (Build Your Own Blocks). Snap is a visual and drag-and-drop programming language. It is a complement of Scratch; the main difference is that it allows you to Build Your Own Blocks and add more complex code. These additional capabilities make it more suitable for computer science to high school (Snap! (Build Your Own Blocks) 4.0, n.d.).

- Greenfoot. Greenfoot teaches object orientation with Java. Greenfoot is visual and interactive. It works with actors that are programmed in standard textual Java code, providing a combination of programming experience in a traditional text-based language with visual execution. Greenroom is an exclusive place to instructors for sharing teaching resources and discussion surrounding teaching with Greenfoot (Greenfoot | About Greenfoot, n.d.).

- Kodu. Kodu lets students create games on Windows PCs via a simple visual programming language. Kodu can be used to teach creativity, problem solving, storytelling, as well as programming. Anyone can use Kodu to make a game, young children as well as adults with no design or programming skills.

- Cubetto. Cubetto is inspired with LOGO Turtle. Cubetto is the friendly wooden robot that teaches the basics of computer programming. It uses a simple programming language that can be touch and manipulated, like LEGO. It is suitable for younger children who cannot know read and write yet (Cubetto: A robot teaching kids code & computer programming, n.d.).

3. Our Proposal

The projects of massification of computational thinking and coding are now starting to be implemented in our education system in Portugal. This is the main reason why most students have never had the opportunity to learn computational thinking or coding.

In this research, a study was conducted to investigate and explore the views of students and the difficulties they faced in learning programming courses. This research also includes the intention to detect the best practices to engage and improve students in learning programing.

Young people, our students, grow up surrounded by technology. But, many of them have no idea how it all works and how important is for their futures. The young people of today have not known life without technology. It is an integral part of their existence. Most of them spend their free time on their computers or their mobiles. They are essential communication and information tools for them. They have grown up with computers and mobiles. For all of this, we don't intend to use technology for this study. It is intentional that students handle and solve the exercises manually, like board games, where they can explore with pleasure, without fear of making mistakes and where teacher-student relationship and confidence can be improved and enhanced.

3.1. Study Group

The study involved a group of students of Computer Engineering from the Polytechnic of Guarda, Portugal. Our study group reveal some general difficulties in the area of CS. From our years of experience, we have found that most of the students have very particular difficulties in terms of computational which affect, in our opinion, the learning programming process.

3.2. Pre-Programming Course (CS0)

Computational thinking may be applied to various kinds of problems that do not directly involve coding tasks. According to the characteristics of the IPG computer engineering students, we have created a set of computational thinking exercises, or activities, to provide students to substantially improve their cognitive abilities. And, consequently, improve programming learning (Figueiredo, Gomes & García-Peñalvo, 2016).

The course session planning activity is:

- Follow and Give instruction. This kind of exercises have as purpose to increase the development of students' cognitive reasoning abilities and spatial visualization, strongly associated with the necessary characteristics for programming (Fincher, et al., 2005; Simon, et al., 2006; Study, 2012). Examples of the proposed exercises are: students should draw on a paper what a student or a teacher describes, and reverse roles; giving directions from point A to point B. Some examples:

Example number 1: Students should design on a paper what a student or a teacher describes. On a sheet of paper draw a square measuring approximately 5 cm on its sides. Draw a small dot in the center of the square. Draw a line that starts at the top right corner to the bottom left corner, passed by the point. Draw a line that starts in the upper left corner to the bottom right corner, passing the point. Write your first name in the triangle below the center point.

Example number 2: It is also possible to practice from an image, asking students to describe it through the design of others images.

- Map Design. With the use of this type of exercises we aim to develop students' capacities in planning, designing and describing in terms of specific characteristics in a concrete situation. These activities include exercises for the student to draw a map of a particular location, or draw a map to go from point A to point B, within our school for example.

- Origami and Paper Folding. Origami and or paper folding (Cooper, Wang, Israni & Sorby, 2015; Falomir, 2016; Jaeger, et al., 2015; Simon, et al., 2006; Study, 2012) is a Japanese secular art widespread throughout the world, known for the development of features, such as: visual and spatial perception, fine motor coordination, memory, patience and persistence, self-confidence, logical thinking, attention, concentration and relieving stress and tension. There are thousands of examples from the simplest to the most complex, of various categories, which can be used according to characteristics and likings of each. In Figure 1, we can see some examples of origami preferred by students.

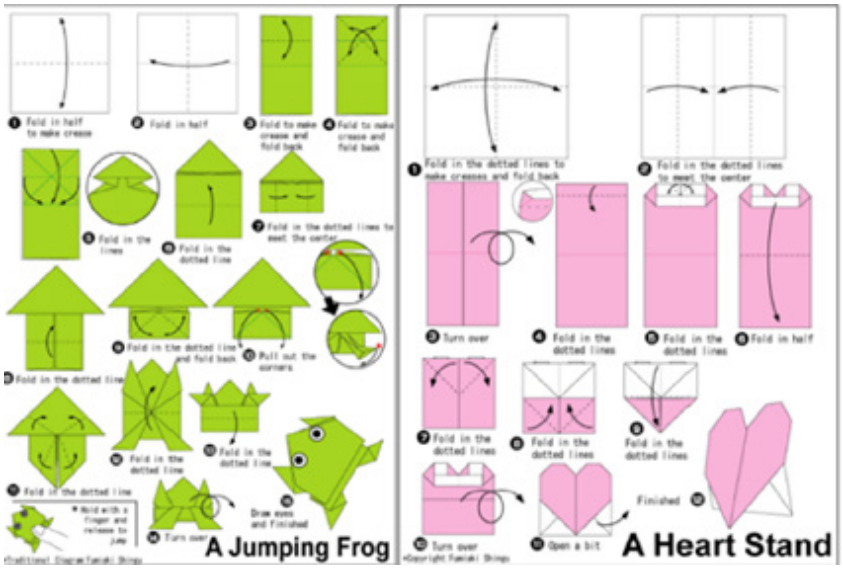

Figure 1. Examples Origami (adapted from http:// http://en.origami-club.com/index.html)

Paper folding, in particular the Punched Holes, is frequently used to investigate the spatial visualization skills. In our case, we want to use this activity for the development of students' capacity by solving various exercises. In this type of exercise students should imagine that they are folding and unfolding paper. In each of the left and right drawing figures there are problems, see Figure 2.

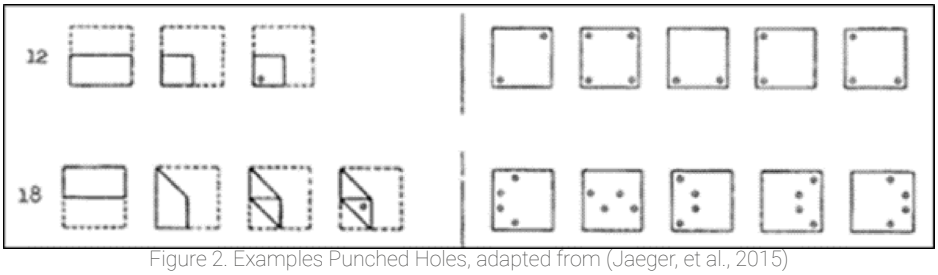

Figure 2. Examples Punched Holes, adapted from (Jaeger, et al., 2015)

- Memory Transfer Language. The used of Memory Transfer Language (MTL) exercises; allows us to overcome some problems detected in the construction of knowledge in early learning programming, particularly in the representation of variables and assignment statements. The methodology used to implement this kind of exercises was based on the representation of instructions in the computer memory (Mselle y Twaakyondo, 2012). Figure 3, are examples of these exercises. The student should use to solve each of the exercises the diagram of Figure 4.

```
Program 1
begin
        int x, y
        read x
        read x, y
        write y, x, y
end

What is the expected output
 if you enter the values 3, 6, 9?
```

```
Program 2
begin
        int x, y
        read x
        read x
        read y
        write y, x, x
end

What is the expected output
 if you enter the values 3, 6, 9?
```

```
Programa 3
begin
        int x, y
        x = 5
        y = x
        x = x + 5
        y = x + 5
        write x, y
end

What is the expected output?
```

Figure 3. Example 1, 2 and 3 for MTL

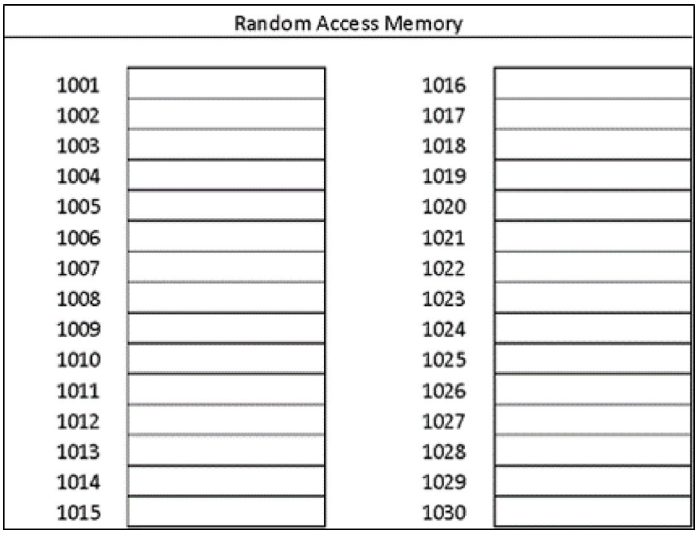

Figure 4. Output representation for MTL exercises

- Parson Problems are assignments for learning programming where the student has to select, order, and indent code fragments (Denny, Luxton-Reilly & Simon, 2008; Ericson, 2014; Morrison, Margulieux, Ericson & Guzdial, 2016), see Figure 5.

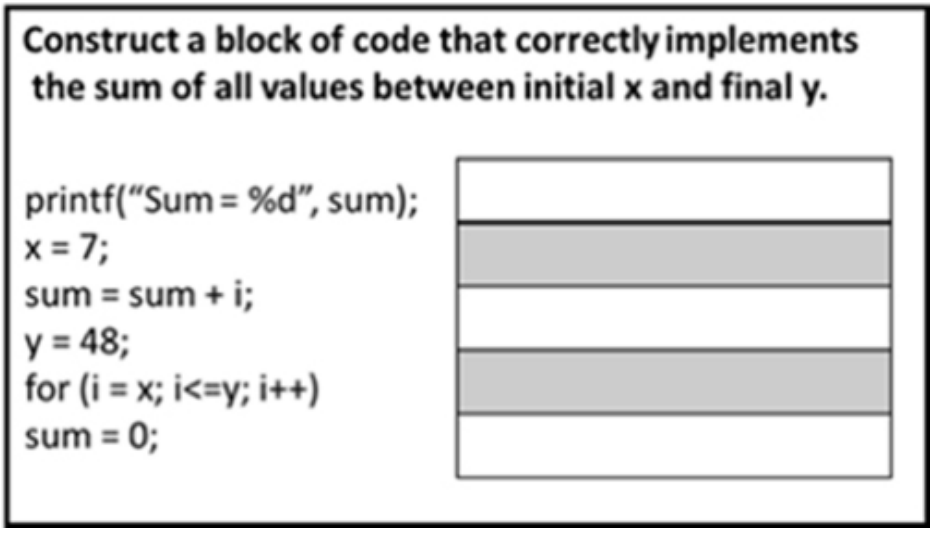

Figure 5. Examples Parson Problems

3.3. Description and Results Study

Two online questionnaires were developed. One with 20 questions about Punched Holes (PH) and another with 5 Follow and Give Instruction (FGI) activities. The number of students who answered the questionnaire was 46.

The PH questionnaire was evaluated by the number of correct answers. The FGI of activities were classified according to the following evaluation scale from not responding (value 0) to detailed description and using references (value 4). Table 1 shows the results PH questionnaire.

		Success	%
PH < 10	9	6	66,7%
PH >= 10	37	25	67,6%
PH >= 14	27	21	77,8%

Table 1. Results PH questionnaire

In the first line, PH <10, 9 of the 46 students had a score of less than 10 correct answers. Of the 9 students, 6 successfully completed the course.

		Success	%
FGI < 3	26	13	50,0%
FGI >= 3	20	18	90,0%

Table 2. Results FGI activities

As it can easily be seen, in Table 2, we highlight the results for FGI>=3, 20 students. Having successfully completed the course 18 students, which corresponds to 90%.

In Table 3, we combine the results of the two activities. It is important to enhance the results combined with the FGI activity, of values greater than or equal to 3. The results are 100%, 88.9% and 100%, for the different PH results. These results lead us to believe that Follow and Give Instruction computational thinking activity has a strong influence on the success of the course.

		Success	%
PH < 10 and FGI < 3	7	4	57,1%
PH < 10 and FGI >= 3	2	2	100,0%
PH >= 10 and FGI < 3	19	9	47,4%
PH >= 10 and FGI >= 3	18	16	88,9%
PH >= 14 and FGI < 3	10	6	60,0%
PH >= 14 and FGI >= 3	15	15	100,0%

Table 3. Results PH and FGI

4. Practice Computational Thinking

Studies have demonstrated the relationship between the style and the level of detail in the description and construction of a map with the objectives of a programming course (Fincher, et al., 2005). According to our results, we can also affirm that there is a strong relationship between the style and the level of detail used for the construction and map design and in the activities of FGI, such as the objectives of the course of introduction to programming.

4.1. Practice Map Design and Follow and Give Instruction

Based on our experience, we intend to implement a set of exercises to work on and study this methodology in order to improve the students' skills in programming and computational thinking.

In these exercises, we will evaluate the level of detail and clarity in the resolution. Some examples:

- Follow and give instructions, or by asking and giving directions, such as the methodology used in language courses, based on maps given to students, such as those in Figure 6 and Figure 7, students should describe in detail the path required to move from point A to point B.

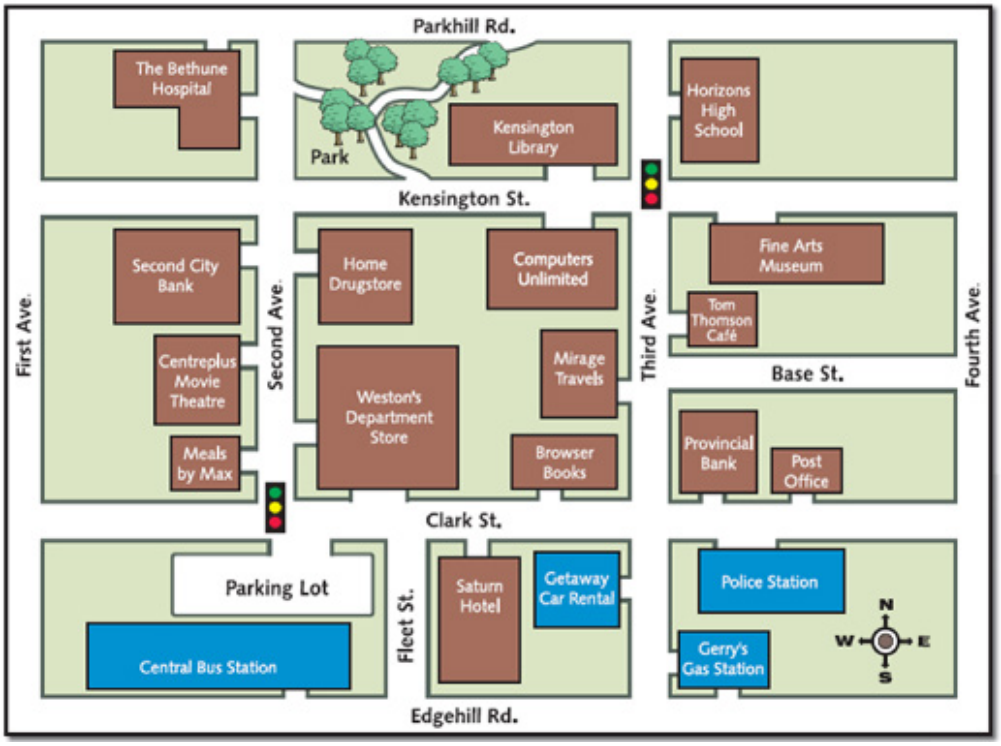

Figure 6. Map 1 to use in exercises of asking and giving directions

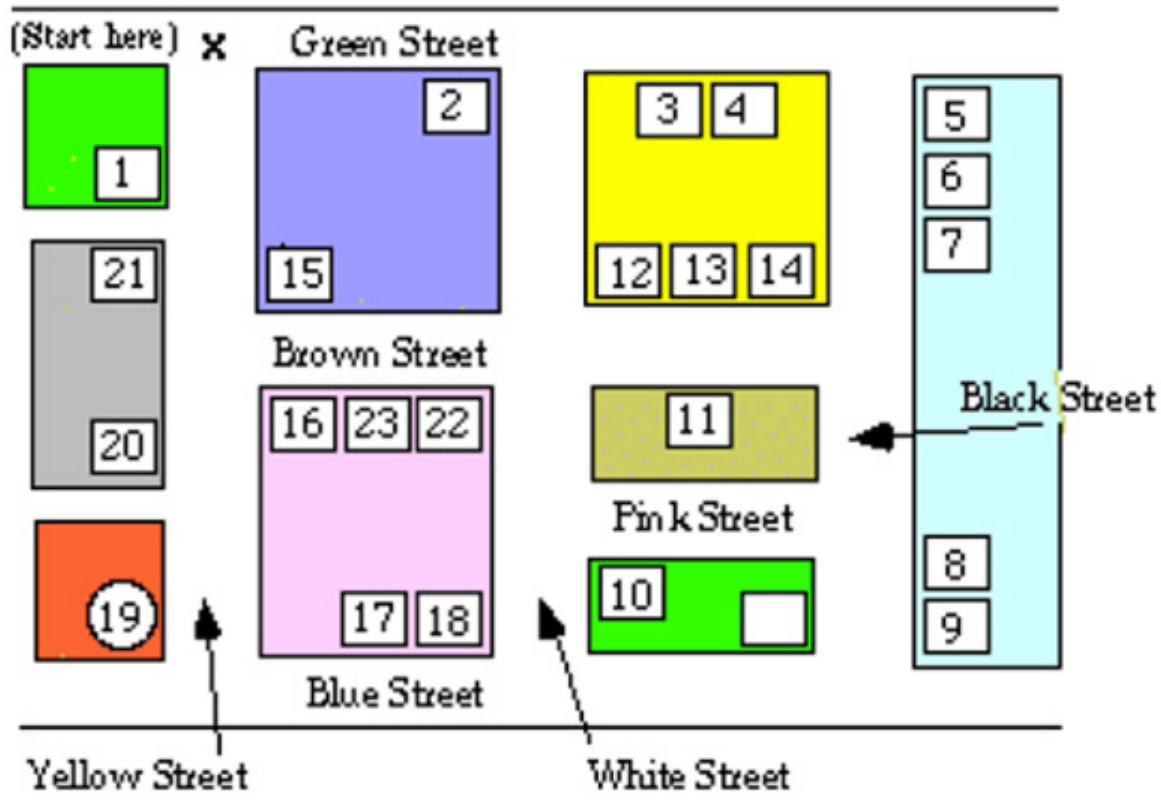

Figure 7. Map 2 to use in exercises of asking and giving directions

- Follow and give instructions should also include exercises where students should draw according to the instructions that a student or a teacher gives, from a drawing or simple text instructions, such as the examples in Figure 8 and Figure 9.

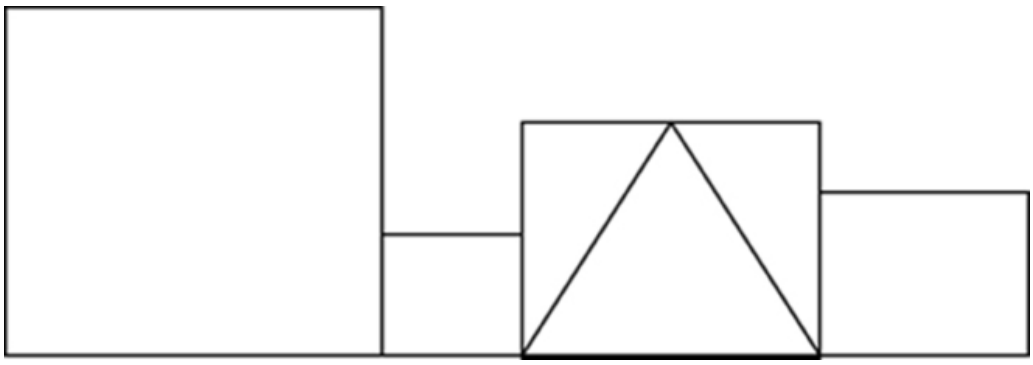

Figure 8. Picture to use with follow and give instructions

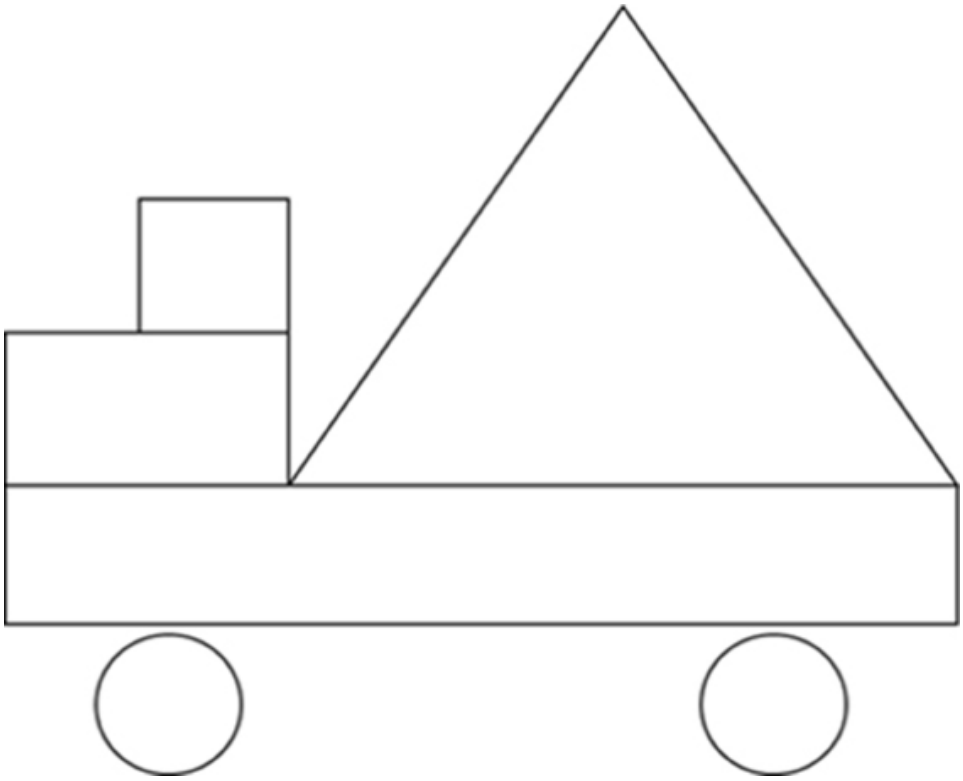

Figure 9. Picture to use with follow and give instructions

4.2. Learning by teaching experience

The advantages of using learning by teaching are well known. People learn more if they need to teach others, suggest the authors of (Nestojko, Bui, Kornell & Bjork, 2014).

Based on these results, we will implement activities where students have to prepare a lesson for others. Those lessons will focus on activities to improve computational thinking and/or programming. They can be used in the "Hour of Code" event. They can also be used for the promotion and development of computational thinking and programming in schools in the region of Guarda made by the students with teachers' supervision.

5. Discussion and Conclusion

In this paper, we have presented many tools that enhance the development of computational thinking skills. However, these tools should be used throughout the learning process, they must be manipulated and explored from 3 to 18 years of age, the usual age of enrolling into higher education (García-Peñalvo, Llorens-Largo, Prieto & Vendrell, 2017; Llorens-Largo, García-Peñalvo, Prieto & Vidal, 2017). The education system in Portugal is now starting to take the first steps in the implementation of computational thinking as a curricular or extracurricular component. Most students of the first year of the Computer Engineering course, from the IPG, mostly did not have the opportunity to develop computational thinking throughout their student life. Hence, the difficulties in learning the initial programming course is a major concern. It is necessary to work these students quickly in an effective way, so that the demotivation does not reach them.

With this work, we intend to present our idea and show the results obtained in our experience with the activities of follow and give instructions. We find the results quite interesting and should, therefore, be explored. To do this, we suggest a set of activities to test with the students. There are many strategies that teachers can employ to increase the skills for students with learning difficulties in computer science. Because computing education, and especially the computational thinking is a new area, many educators may not know how to provide support to students as they learn computing. In this article, we suggest some strategies and resources that educators can implement to support students.

6. References

Alice – Tell Stories. Build Games. Learn to Program. (n.d.). Retrieved July 5, 2017, from http://www.alice.org/

App Inventor for Educators – MIT App Inventor Educators Community. (n.d.). Retrieved July 6, 2017, from http://teach.appinventor.mit.edu/

Blockly | Google Developers. (n.d.). Retrieved July 6, 2017, from https://developers.google.com/blockly/

CodeCombat - Learn how to code by playing a game. (n.d.). Retrieved July 5, 2017, from https://codecombat.com/

Coding for Kids | Tynker. (n.d.). Retrieved July 5, 2017, from https://www.tynker.com/

Computer Science Education Week. (n.d.). Retrieved July 5, 2017, from https://csedweek.org/

Computer Science Unplugged. (n.d.). Retrieved July 5, 2017, from http://csunplugged.org/

Computing At School. (n.d.). Retrieved July 6, 2017, from http://www.computingatschool.org.uk/

Cooper, S., Wang, K., Israni, M. & Sorby, S. (2015). Spatial Skills Training in Introductory Computing. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, 13-20. doi:http://doi.org/10.1145/2787622.2787728

Cubetto: A robot teaching kids code & computer programming. (n.d.). Retrieved August 6, 2017, from https://www.primotoys.com/

Denny, P., Luxton-Reilly, A. & Simon, B. (2008). Evaluating a new exam question: Parsons problems. *Proceedings of the Fourth International Workshop on Computing Education Research*, 113-124. http://doi.org/10.1145/1404520.1404532

Ericson, B. J. (2014). Adaptive Parsons Problems with Discourse Rules. *Icer '14*, 145-146. http://doi.org/10.1145/2632320.2632324

Falomir, Z. (2016). Towards A Qualitative Descriptor for Paper Folding Reasonin. In *Proc. of the 29th International Workshop on Qualitative Reasoning (QR'16)*. New York, USA.

Figueiredo, J., Gomes, N. & García-Peñalvo, F. J. (2016). Ne-course for learning programming. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM '16* (pp. 549–553). New York, New York, USA: ACM Press. doi:http://doi.org/10.1145/3012430.3012572

Fincher, S., Baker, B., Box, I., Cutts, Q., Raadt, M. De, Haden, P., … Tutty, J. (2005). Computer Science at Kent programming courses, (1).

García-Peñalvo, F. J. (2016a). A brief introduction to TACCLE 3 – Coding European Project. In F. J. García-Peñalvo & J. A. Mendes (Eds.), *2016 International Symposium on Computers in Education (SIIE 16)*. USA: IEEE. doi:http://doi.org/10.1109/SIIE.2016.7751876

García-Peñalvo, F. J. (2016b). Proyecto TACCLE3 – Coding. In F. J. García-Peñalvo & J. A. Mendes (Eds.), *XVIII Simposio Internacional de Informática Educativa, SIIE 2016* (pp. 187-189). Salamanca, España: Ediciones Universidad de Salamanca.

García-Peñalvo, F. J. (2016c). What Computational Thinking Is. *Journal of Information Technology Research, 9*(3), v-viii.

García-Peñalvo, F. J., & Cruz-Benito, J. (2016). Computational thinking in pre-university education. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing*

Multiculturality - TEEM '16 (pp. 13-17). New York, New York, USA: ACM Press. doi:http://doi.org/10.1145/3012430.3012490

García-Peñalvo, F. J., Hughes, J., Rees, A., Jormanainen, I., Toivonen, T., Reimann, D., . . . Virnes, M. (2016). *Evaluation of existing resources (study/analysis)*. Belgium: TACCLE3 Consortium. doi:http://doi.org/10.5281/zenodo.163112

García-Peñalvo, F. J., Llorens Largo, F., Molero Prieto, X. & Vendrell Vidal, E. (2017). Educación en Informática sub 18 (EI<18). *ReVisión, 10*(2), 13-18.

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A. & Jormanainen, I. (2016). *An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers*. Belgium: TACCLE3 Consortium. doi:http://doi.org/10.5281/zenodo.165123.

Greenfoot | About Greenfoot. (n.d.). Retrieved July 7, 2017, from https://www.greenfoot.org/overview

Home - Barefoot Computing Barefoot Computing. (n.d.). Retrieved July 5, 2017, from https://barefootcas.org.uk/

Israel, M., Wherfel, Q. M., Pearson, J., Shehab, S. & Tapia, T. (2015). Empowering K-12 Students with Disabilities to Learn Computational Thinking and Computer Programming. *TEACHING Exceptional Children, 48*(1), 45-53. doi: http://doi.org/10.1177/0040059915594790

Jaeger, A. J., Wiley, J., Pellegrino, J., Zinsser, K., Stieff, M. & Moher, T. (2015). *What Does the Punched Holes Task Measure?*

Khan Academy | Free Online Courses, Lessons & Practice. (n.d.). Retrieved July 5, 2017, from https://www.khanacademy.org/

Lightbot. (n.d.). Retrieved July 5, 2017, from https://lightbot.com/flash.html

LiveCode Ltd. (n.d.). LiveCode in Education | LiveCode. Retrieved July 6, 2017, from https://livecode.com/products/livecode-platform/livecode-in-education/

Llorens Largo, F., García-Peñalvo, F. J., Molero Prieto, X. & Vendrell Vidal, E. (2017). La enseñanza de la informática, la programación y el pensamiento computacional en los estudios preuniversitarios. *Education in the Knowledge Society, 18*(2), 7-17. doi: http://doi.org/10.14201/eks2017182717

Microsoft Touch Develop - create apps everywhere, on all your devices! (n.d.). Retrieved July 6, 2017, from https://www.touchdevelop.com/

MIT App Inventor. (n.d.). Retrieved July 5, 2017, from http://ai2.appinventor.mit.edu/

Morrison, B. B., Margulieux, L. E., Ericson, B. & Guzdial, M. (2016). Subgoals Help Students Solve Parsons Problems. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 42-47. http://doi.org/10.1145/2839509.2844617

Mselle, L. J. & Twaakyondo, H. (2012). The impact of Memory Transfer Language (MTL) on reducing misconceptions in teaching programming to novices. *International Journal of Machine Learning and Applications, 1*(1), 1-6. doi:http://doi.org/10.4102/ijmla.v1i1.3

National Research Council. (2012). *A Framework for K-12 Science Education. Practices, Crosscutting Concepts, and Core Ideas.* Washington, D.C.: National Academies Press. doi:http://doi.org/10.17226/13165

Nestojko, J. F., Bui, D. C., Kornell, N. & Bjork, E. L. (2014). Expecting to teach enhances learning and organization of knowledge in free recall of text passages. *Memory & Cognition, 42*(7), 1038-1048. doi:http://doi.org/10.3758/s13421-014-0416-z

Pinto-Llorente, A. M., Casillas-Martín, S., Cabezas-González, M. & García-Peñalvo, F. J. (2017). Building, coding and programming 3D models via a visual programming environment. *Quality & Quantity, In Press*. doi:http://doi.org/10.1007/s11135-017-0509-4

Programming for Kids | Kodable. (n.d.). Retrieved July 5, 2017, from https://www.kodable.com/

Scratch - Imagine, Program, Share. (n.d.). Retrieved July 5, 2017, from https://scratch.mit.edu/

Simon, Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q., … Tutty, J. (2006). Predictors of success in a first programming course. *Proceedings of the 8th Austalian Conference on Computing Education - Volume 52*, 189-196. doi:http://doi.org/10.1145/953051.801357

Snap! (Build Your Own Blocks) 4.0. (n.d.). Retrieved July 10, 2017, from http://snap.berkeley.edu/index.html

STEM. (n.d.). Retrieved July 6, 2017, from https://www.stem.org.uk/

Study, N. E. (2012). An Overview of Tests of Cognitive Spatial Ability. 66th EDGD Mid-Year C*onference Proceedings*. Retrieved from https://goo.gl/YwnYrv

Taccle 3 – Supporting primary teachers to teach coding. (n.d.). Retrieved July 5, 2017, from http://www.taccle3.eu/en/

techliteracy. (n.d.). Retrieved July 6, 2017, from https://techliteracy.co.uk/

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. doi:http://doi.org/10.1145/1118178.1118215